15 Sep 2023 Parallel Algorithms, Algebraic Branching Programs

Announcements.

1. HW2 "re-matching questionnaire" : see Ed for link. Fill out by 20:00 Sunday to re-match.
2. HW2 is on Gradescope. Due in 2 weeks, again 4 problems.

Recall: Want to add binary numbers expressed by bit strings $a, b \in \{0,1\}^n$, using parallel algorithm.

This reduces to computing the "carry bits"

$$c_i := \begin{cases} 1 & \text{if the numbers encoded by } a_i \ldots a_n \text{ and } b_i \ldots b_n \text{ sum up to } 2^{n-i+1} \text{ or greater.} \\ \emptyset & \text{otherwise.} \end{cases}$$

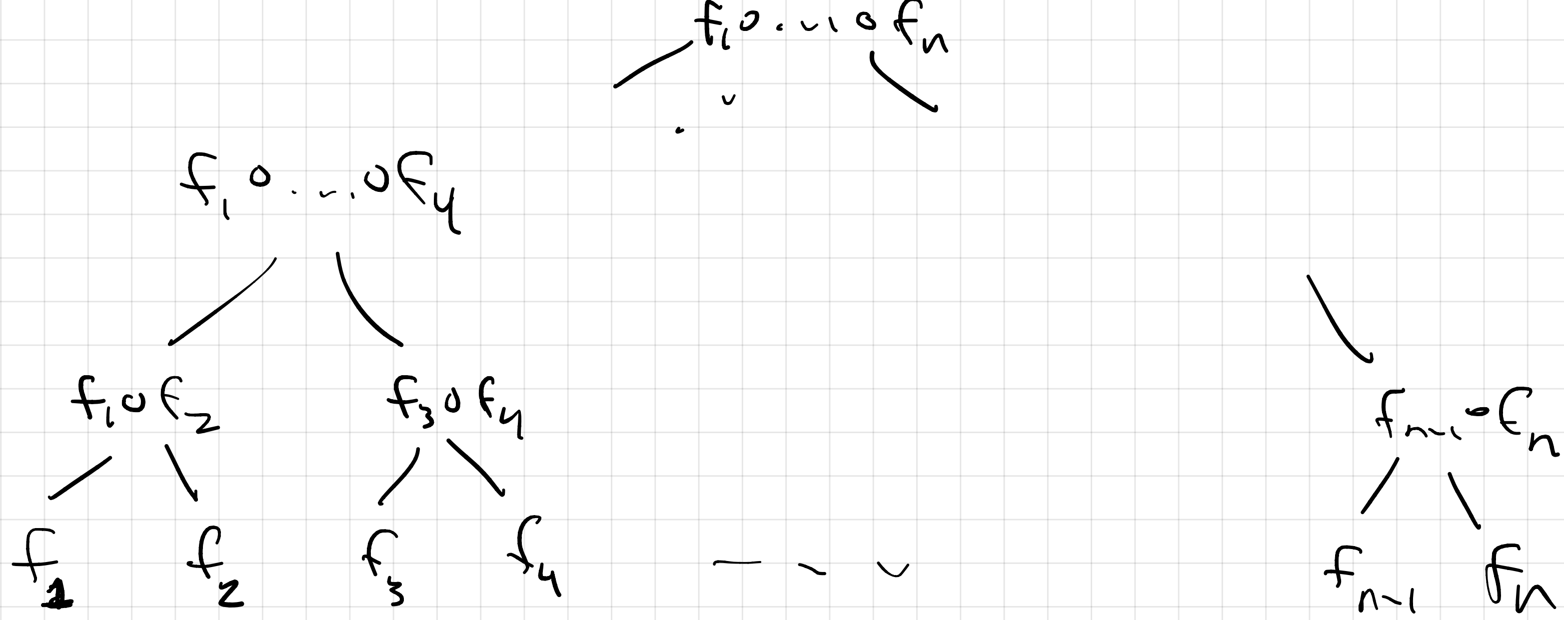
+ $\Sigma = \{\emptyset, 1, *\}$ and let $\mathcal{F}$ be the following set of 3 functions $\Sigma \to \Sigma$.

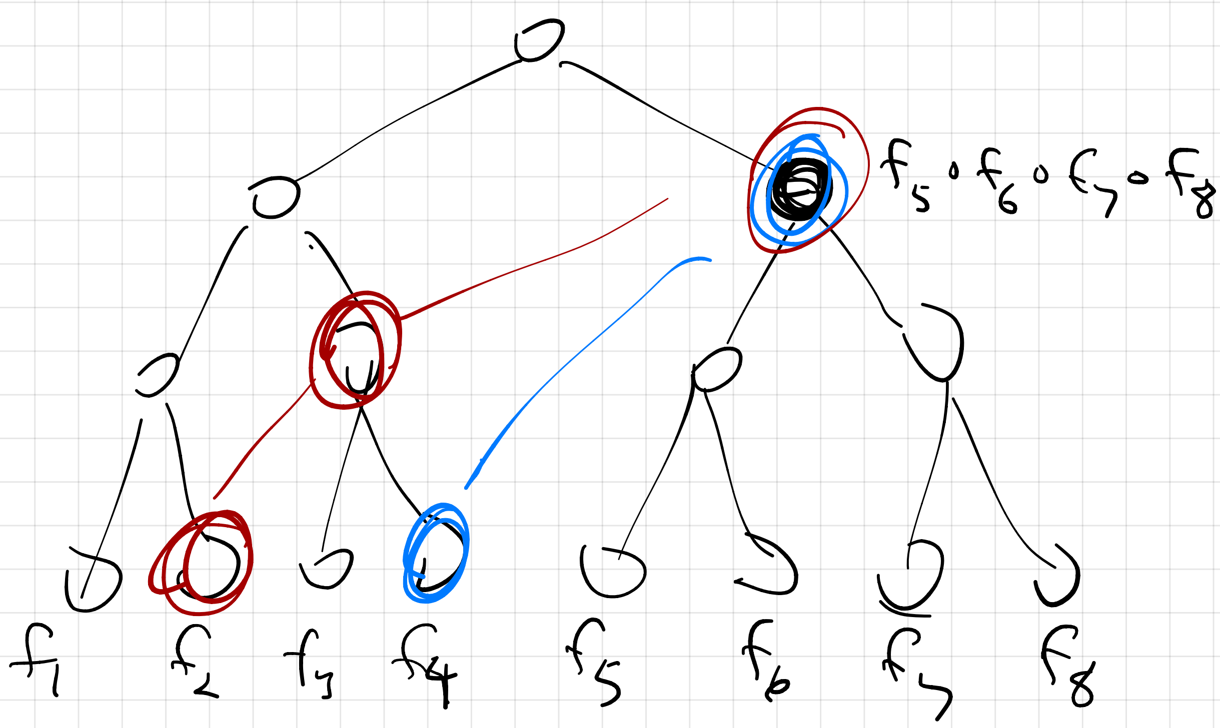
- g_0 = constant function $\emptyset$
- g_1 = " " 1
- i = identity function.

~~Note~~ $\mathcal{F}$ closed under function composition.

Then set $f_i = \begin{cases} g_0 & \text{if } a_i = b_i = 0 \\ i & \text{if } a_i + b_i = 1 \\ g_1 & \text{if } a_i = b_i = 1 \end{cases}$

Then $c_i = f_{i+1} \circ f_{i+2} \circ \ldots \circ f_n(0)$





Step 1. Compute the (2-bit) representation of the function at each node of this tree.

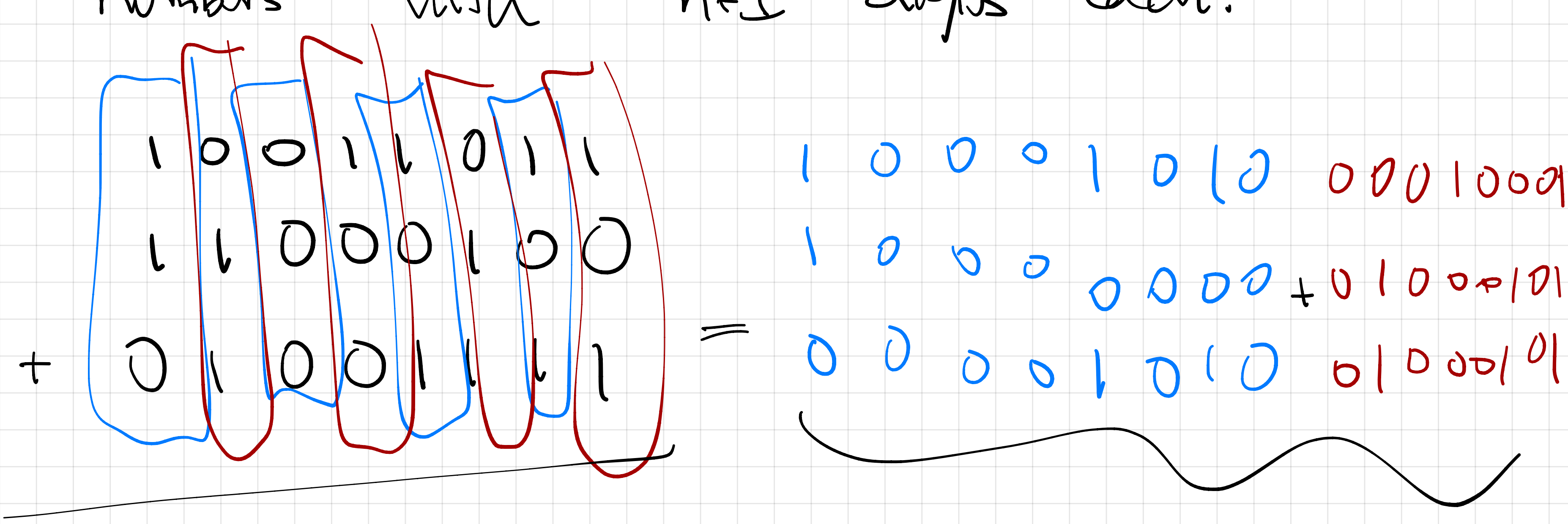
$O(n)$ work, $O(\log n)$ depth

Step 2. In parallel for each i compute $f_{i+1} \circ f_{i+2} \circ \cdots \circ f_n(0)$ in $O(\log n)$ work, $O(\log \log n)$ depth.

Total: $O(n \log n)$ work, $O(\log n)$ depth.

Integer mult: Compute partial products of two binary numbers in $O(n^2)$ work, $O(1)$ depth. Now mult. reduces to adding n numbers of $\leq 2n$ bits.

There's a $O(n)$ work, $O(1)$ depth reduction from adding 3 binary numbers with n digits each to adding 2 binary numbers with $n+1$ digits each.



Add without carries

Blue sum + Red sum

$(\leq n+1$ digits$)$

After $\log_{3/2}(n)$ rounds of reorganizing, we get just 2 numbers, each with $\leq n + \log_{3/2}(n)$ digits, and add them using preceding algorithm.

Work: $O(n^2 + n \log n)$

Depth: $O(\log n)$

<u>Matrix multiplication</u>: Two $n \times n$ matrices with entries of $\leq O(n)$ bits, just compute n^2 dot products in parallel.

Each dot product is

n mult's that can be done in parallel

$O(n^2 \log n)$ work $\quad$ $O(\log n)$ depth

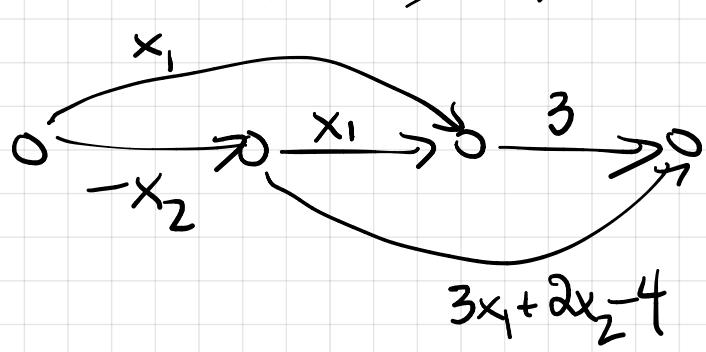
followed by adding n numbers of $O(n)$ bits

$O(n \log n)$ work $\quad$ $O(\log n)$ depth

<u>Total</u>: $O(n^4 \log n)$ work, $\quad$ $O(\log n)$ depth.

<u>Algebraic Branching Programs.</u> An ABP is a diagram consisting of a directed acyclic graph (DAG) $G = (V, E)$, with $V = \{v_0, v_1, \ldots, v_{n-1}\}$, with every edge $(v_i, v_j) \in E$ satisfying $v_i < v_j$.

Plus, each edge is labeled with a linear form (degree-1 polynomial) in variables $x_1, \ldots, x_m$.



$f^{\pi} = 3x_1 - 3x_1x_2 - 3x_1x_2 - 2x_2^2 + 4x_2$

If π is an ABP, the polynomial f^{π} is the polynomial

$$f^{\pi}(x_1, \ldots, x_m) = \sum_{\substack{P \text{ path} \\ \text{from } v_0 \text{ to } v_{n-1}}} \prod_{e \in P} \text{label}(e)$$